目录

[HSF 服务框架主要组件 2](#_Toc3194441)

[1、服务提供者 2](#_Toc3194442)

[2、服务调用者 2](#_Toc3194443)

[3、地址服务器 3](#_Toc3194444)

[4、配置服务器 3](#_Toc3194445)

[5、Diamond 服务器 3](#_Toc3194446)

[1、HSF 框架采用 Netty + Hession 数据序列化协议实现服务交互 6](#_Toc3194447)

[2、HSF 框架的容错机制 7](#_Toc3194448)

[3、HSF 框架的线性扩展支持 8](#_Toc3194449)

阿里巴巴集团内部使用的分布式服务框架 HSF（High Speed Framework，也有人戏称“好舒服”）已经被很多技术爱好者所熟知，目前已经支撑着近 2000 多个应用的运行。

其对应早期的开源项目 Dubbo（因为某些原因，Dubbo 项目在 2012 年年底，阿里巴巴就停止了对此开源项目的更新），则更是在互联网领域有着非常高的知名度和广泛的使用。

本文通过对阿里巴巴 HSF 服务框架的介绍，让大家能对这类分布式服务框架架构设计、运行原理，以及如何实现作为一个 SOA 架构需要满足的各个特性有一个清晰的认识。

## **HSF 服务框架主要组件**

### **1、服务提供者**

在服务框架中真正提供服务功能实现的应用实例，为了保障服务提供的高可用性，一般均是集群部署。

每一个 HSF 的应用均是以 War 包的形式存在，运行在阿里巴巴优化定制后的 Tomcat 容器中，在 Tomcat 容器层已经集成了 HSF 服务框架对服务提供者或服务调用者进行配置服务器发现、服务注册、订阅、失效转移等相关功能，所以不管是在服务提供者还是调用者开发时，只需要进行服务相关的配置操作，应用中无需引入任何 HSF 相关的 Jar 依赖包。

考虑到应用故障的隔离、更方便的服务管控，**目前淘宝内部大部分应用的部署方式还是一个虚拟机（对应一个操作系统）运行一个 Tomcat 容器**，每个 Tomcat 运行一个服务应用，随着近几年以 Docker 容器技术的发展和流行，现在阿里巴巴内部也正在进行应用容器化部署的工作，让服务器的资源利用更加科学和高效。

### **2、服务调用者**

作为服务的消费者，大多数也是以 WAR 应用包的方式运行在 Tomcat 容器中，在阿里巴巴集团内部也有一部分是基于 C/C++、PHP、Node.js 等语言开发的服务调用者。

### **3、地址服务器**

在 HSF 服务框架中，地址服务器肩负着给服务提供者和服务调用者提供部署环境中所有配置服务器和 Diamond 服务器的服务器列表信息，是由 Nginx（ 是一个高性能的 HTTP 和反向代理服务器）提供该服务能力。

在部署 HSF 服务环境时，会将整个环境中的配置服务器集群（服务器 IP 列表）和 Diamond 服务器集群信息设置在地址服务器上，在实际生产部署中，也会部署多台地址服务器提供负载均衡和高可用性的服务，服务提供者和调用者**通过统一域名的方式访问**这些地址服务器（比如“xxx.tbsite.net”），通过 DNS 轮询实现地址服务器访问的高可用性。

### **4、配置服务器**

配置服务器在 HSF 框架中主要负责记录环境内所有服务发布（服务提供者的 IP 地址和服务端口信息）和服务订阅（服务调用者的 IP 地址和服务端口信息）信息，并将服务相关信息推送到服务节点上。为了追求服务发布和订阅的推送效率，所有的服务发布和订阅信息均是保存在内存中。

**配置服务器与所有服务者提供者和调用者均是长连接，采用心跳的方式可监控到各服务运行节点的状况**，一旦出现服务提供者服务节点出现故障时，会自动推送更新后（将出问题的服务提供者服务节点信息从列表中删除）的服务提供者列表给相关的服务调用者端。

在生产环境中，会部署多台配置服务器，用于服务发布、订阅、推送的负载均衡，在多台配置服务器间会进行实时的数据同步，保证服务发布和订阅信息尽快能同步到各服务节点上。

某种程度上，配置服务器在 HSF 框架中扮演了服务调用调度的指挥官，通过给服务调用者端推送不同的服务提供者列表就可以轻易的调整服务调用的路由，这一特性在淘宝平台实现单元化（即某一客户在访问淘宝时，访问请求一旦路由到某一个淘宝机房后，在淘宝上进行的所有业务的操作均可以在该机房完成，而无需访问其他机房的服务，也是实现异地多活的基础）、异地多活起到了至关重要的作用。

### **5、Diamond 服务器**

本质上，**Diamond 服务器是一个通用的统一配置管理服务**，类似 ZooKeeper，给应用提供统一的配置设置和推送服务，使用场景非常广泛，在阿里巴巴内部有很多的产品在需要进行配置的保存和获取时都会使用 Diamond 服务器。

在 HSF 服务框架中，则主要承担了服务调用过程中对于服务调用安全管控的规则、服务路由权重、服务 QPS 阀值等配置规则的保存，**所有的信息均是持久化保存到了后端的 MySQL 服务器中**，在生产环境中，会有多台 Diamond 服务器提供负载均衡的服务。

使用 Diamond 服务器进行服务相关设置的典型场景如下：

* 通过设置白名单（服务调用者所在服务节点 IP 地址）的方式设置某些服务或服务中的方法只能让特定 IP 地址的服务器调用；
* 通过用户认证的方式控制服务是否能够调用；
* 按照不同的服务器权重设置服务调用者对多个服务提供者服务节点的访问；
* 设置某些服务的 QPS 能力上限值，一旦该服务的 QPS 达到该阀值，则拒绝服务的继续调用，这也是实现服务限流的技术实现，在平台进行大促或秒杀场景时，保障平台的稳定的重要屏障。

通过这样规则的设置，Diamond 除了将这些规则保存在自身的数据库中外，会自动将这些规则推送到相关的服务节点上（实际实现上是服务节点会定时从 Diamond 服务器上同步相关配置信息），使这些规则能立即在服务运行环境中生效。

如图 3-5 所示是阿里巴巴 HSF 服务框架的工作原理，按照服务注册发布、服务订阅、服务规则推送、最终服务提供者和服务调用者间的服务交互的顺序说明了 HSF 服务框架中每个组件在整个框架中所扮演的角色。

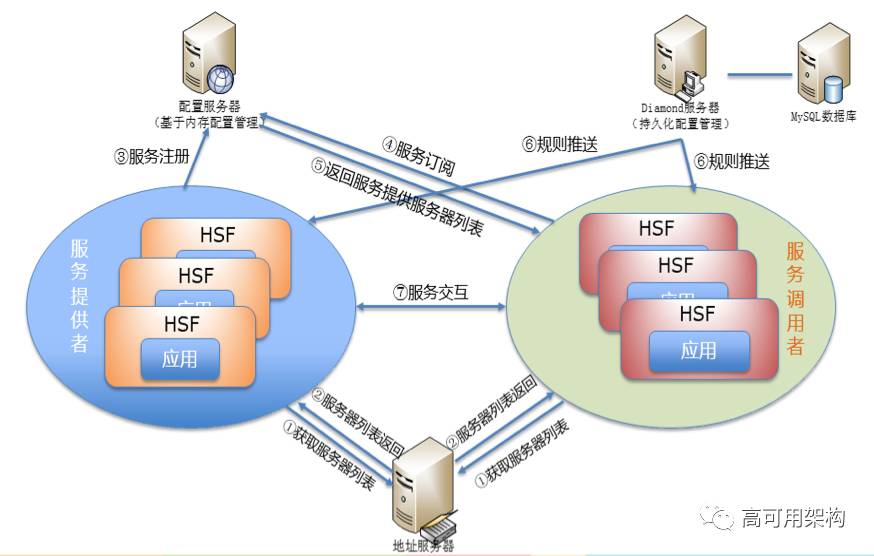


图 3-5 HSF 服务框架工作原理示意图

#### **1）服务节点对配置服务器列表的获取。**

服务调用者和服务提供者在随着 Tomcat 容器启动后，会以域名（比如“xxx.tbsite.net”）的方式获取到可用的地址服务器，通过向地址服务器分别发送获取配置服务器和 Diamond 服务器服务 IP 列表请求的方式，在容器启动完成后，就已经在该服务节点上获取到了配置服务器和 Diamond 服务器的 IP 列表信息。整个过程如图 3-5 中的步骤①②。

#### **2）服务的注册发布。**

作为服务提供者，当获取到配置服务器的服务器列表后，则向配置服务器发送当前应用中包含的服务提供者相关信息（这些信息均是从应用的配置文件中获取到，比如服务的接口类全名、服务版本、所属服务组等信息），联同当前服务器的 IP 地址、服务端口等信息进行服务注册发布，如图 3-5 中的步骤③。这个步骤在每一个有服务提供的应用启动时都会自动执行，比如现在有 5 个提供同一服务的应用启动后，此时在配置服务器上就已经保存了提供这一服务的 5 个服务器相关信息。

#### **3）服务的订阅。**

当作为服务调用者的应用启动时，同样在完成配置服务器列表的获取后，就进行与配置服务器的交互，发送服务消费者相关信息（同样包含了服务的接口全名，服务版本、所属服务组）到配置服务器进行服务的订阅，此时在配置服务器上会通过服务接口全名+服务版本作为匹配条件在当前配置服务器的内存中进行搜索，一旦获取到对应的服务注册信息，则将对应的服务提供者的服务器组 IP 地址及端口返回给服务调用者所在的应用节点上，此时也就完成了服务调用者端对于它所需要调用的服务提供者服务器列表信息，用于在服务真正交互时使用。服务订阅过程如图 3-5 中的步骤④⑤。

#### **4）服务规则的推送（如果需要）。**

如果没有上文提到对于服务安全管控、流量控制等需求的时候，对于 Diamond 服务器的使用并不是必需的，在有这样的需求场景时，可通过 Diamond 提供的规则设置界面，可以对指定服务的服务提供者和调用者设置相关的规则，一旦保存规则后，则此规则配置将会在 5 秒内推送到与所设置服务相关的服务节点上。如图 3-5 中的步骤⑥。

#### **5）服务交互。**

在应用进行业务请求处理过程中，出现了服务调用者对服务提供者的调用时，服务调用者会从已经保存在该应用节点上的服务提供者服务器列表中选择（阿里巴巴内部使用随机模式）其中一台进行服务请求的发送，服务交互期间完全是服务调用者和服务提供者间两台服务器间的，无需通过中间服务器的中转，这就是相比于“中心化” ESB 模式，所有服务交互都需要“中心” ESB 进行服务路由，而当前这种架构称为“去中心化”的主要原因。如图 3-5 中的步骤⑦。

阿里巴巴的分布式服务框架核心是以服务化的方式构建整个应用体系的同时，要保证在高并发的情况下，服务具备高效交互、高可用性和扩展能力。接下来对于 HSF 框架如何给服务提供以上能力具体加以说明。

### **1、HSF 框架采用 Netty + Hession 数据序列化协议实现服务交互**

HSF 框架中采用如今流行的网络通信框架 Netty 加上 Hession 数据序列化协议实现 HSF 服务间的交互，主要考虑点是在大并发量时，服务交互性能达到最佳。这类 RPC 协议采用多路复用的 TCP 长连接方式，在服务提供者和调用者间有多个服务请求同时调用时会共用同一个长连接，即一个连接交替传输不同请求的字节块。它既避免了反复建立连接开销，也避免了连接的等待闲置从而减少了系统连接总数，同时还避免了 TCP 顺序传输中的线头阻塞（head-of-line blocking）问题。

Hessian 是 HSF 框架中默认使用的数据序列化协议，在数据量较小时性能表现出众，Hessian 的优点是精简高效，同时可以跨语言使用，目前支持 Java, C++, .net, Python, ruby 等语言。另外 Hessian 可以充分利用 Web 容器的成熟功能，在处理大量用户访问时很有优势，在资源分配、线程排队、异常处理等方面都可以由 Web 容器保证。

HSF 框架同时也支持切换使用 Java 序列化，Hession 相比 JDK 标准的序列化方式（即基于 Serializable 接口的标准序列化），在典型场景中，其序列化时间开销可能缩短 20 倍。虽然 Hessian 不是最快的序列化协议，但它对于复杂业务对象的序列化正确率、准确性相较于最稳定的 Java 序列化并不逊色太多。

业界还有一些比 Hessian 更快的序列化协议，但它们相对于 Hessian 在复杂场景下的处理能力还是会差一些，**所以 Hessian 是在性能和稳定性同时考虑下最优的序列化协议。**

阿里巴巴当时在对多种通信协议和数据序列化组件等测试中，Netty + Hession 的组合在互联网高并发量的场景下,特别是在 TPS 上达到 10w 以上时，性能和效率远比 REST 或者 Web Service 高。

### **2、HSF 框架的容错机制**

因为要保证服务的高可用性，所以在生产环境部署中一定会有**多个应用实例作为服务提供者提供某一相同服务。**

基于之前所提到的服务框架的运行原理的说明，在进行服务调用时，服务调用者端已经保存了它所需要调用的服务提供者的服务器列表信息（如图 3-6 中为例，则保存了三台服务提供者所在服务器的列表）。

当采用随机方式获取其中一台进行服务交互时（如图 3-6 步骤①），不管是第一台服务器已经某种故障造成了服务请求无法响应，还是该服务器已经接收了服务请求，在进行服务请求处理过程中出现了服务器故障（比如宕机、网络问题），造成该服务器没有在规定的时间（一般服务调用会设置到期时间）返回服务处理的结果，服务调用者端则会获取到服务调用失败的反馈（如图 3-6 步骤②）。

在 HSF 服务调用的代码中会立即从剩下的服务提供者服务器列表中选择另外一个服务器再次进行服务请求（如图 3-6 步骤③），这一次这个服务提供者实例正常提供了此次服务的请求（如图 3-6 步骤④），从而保证了在个别服务提供者出现故障时，完全不会影响该服务正常提供服务。

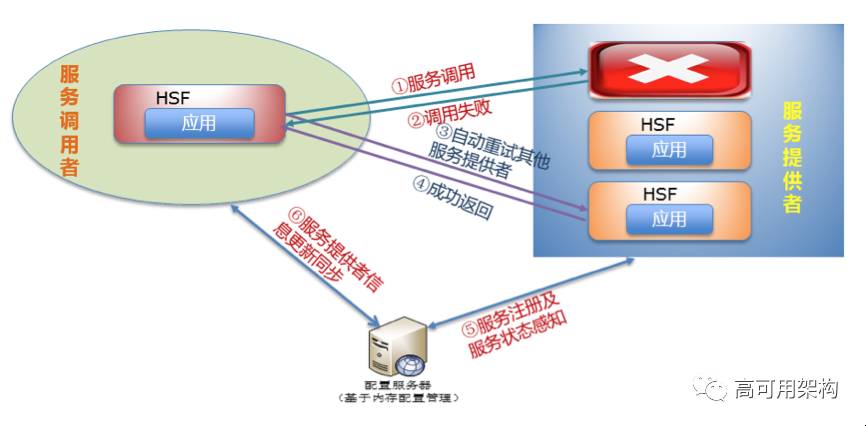


图 3-6 HSF 服务框架实现服务高可用性原理示意图

同时，**因为配置服务器是采用长连接的方式与服务节点进行网络通讯，一旦发现有服务提供者实例出现故障，配置服务器在秒级就会感知到**（如图 3-6 步骤⑤），此时会将出问题这台服务提供者的信息从该服务的服务器列表中删除，并将更新后的服务器列表采用推送的方式同步给与该服务相关的所有服务调用者端（如图 3-6 步骤⑥），这样当下次服务调用者再进行此服务的调用时，就不会因为随机的方式再次对已经停止服务提供的服务器发起服务的调用。

### **3、HSF 框架的线性扩展支持**

作为 HSF 框架设计之初，**最为重要的一个特性就是服务能力的可扩展性。**也就是真正的做到某个服务的业务处理能力能随着服务器资源的增加得到线性的增长。

其实在传统架构中一直也会强调平台的扩展能力，但均会程度不一的出现服务节点数量到达一定量后，出现阻碍平台服务能力扩展的问题，有的是出现网络传输的瓶颈、也有服务节点接入数量上的限制，前文所描述的 ESB 架构带来的“雪崩”效应也均是这类架构给服务能力的扩展带来影响的原因所在。

如图 3-7 中所描述的场景，当服务面对较大的服务调用压力或将要面临如天猫双11大促、秒杀等活动前，已有的服务提供者各服务器水位（CPU、内存、IO等）处于比较高的情况或现有服务能力满足不了业务访问量的要求时，则需要通过增加服务节点数量的方式提升该服务的服务处理能力。

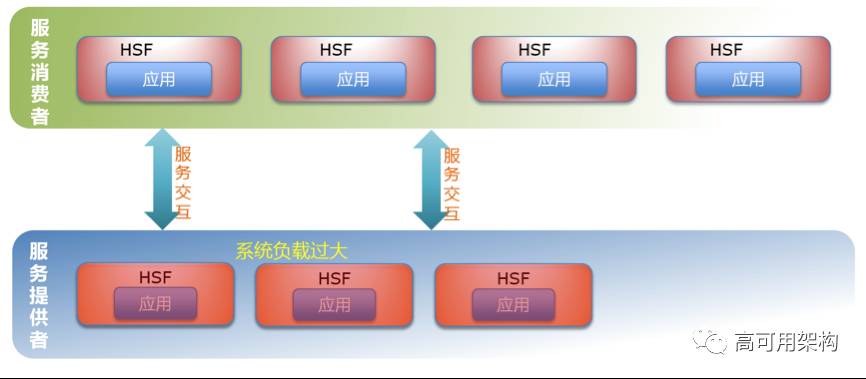


图 3-7 HSF 服务框架对服务能力线性扩展支持1

此时，只需要通过增加该服务的服务提供者实例（如图 3-8 所示，增加了一个服务），基于 HSF 框架的运行机制，**新增加的服务提供者实例一旦应用启动完成后，可在几秒内开始进行服务请求的处理**（主要完成服务注册发布、更新后服务列表推送到服务调用者端），从而达到分担其他服务器实例压力的作用，实现服务能力整体水位恢复到正常的状态（如图 3-9）。

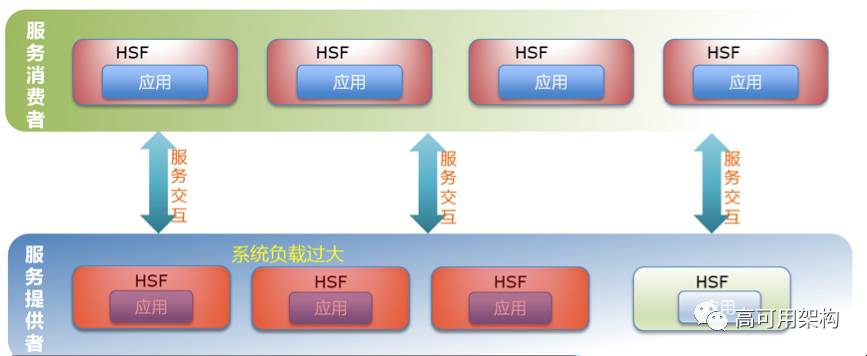


图 3-8 HSF服务框架对服务能力线性扩展支持2

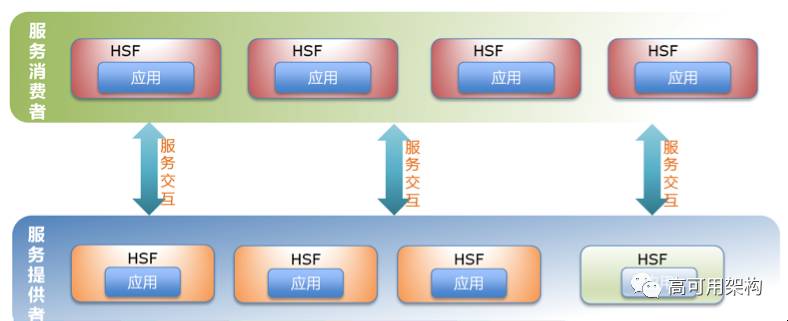


图 3-9 HSF 服务框架对服务能力线性扩展支持3

正是基于 HSF 框架这一特性，从而真正实现了只要增加服务实例就能实现该服务能力扩展的目标，目前在阿里巴巴共享服务事业部中多个服务中心在天猫双 11 那天各自所部署的服务实例节点数量均超过 2000，即同一个服务由超过 2000 个服务实例同时提供负载均衡的服务。